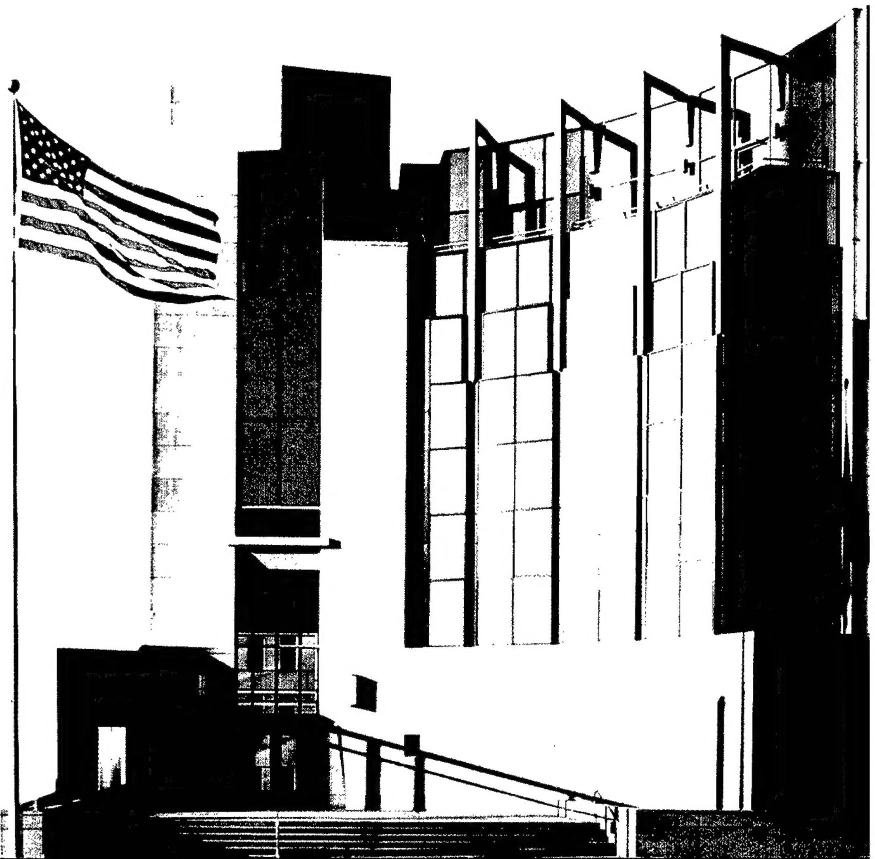
Carnegie Mellon
Software Engineering Institute

Fifth Product Line Practice Workshop Report

Paul Clements
Patrick Donohoe
Kyo Kang
John McGregor
Linda Northrop

September 2001

TECHNICAL REPORT
CMU/SEI-2001-TR-027
ESC-TR-2001-027



Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Fifth Product Line Practice Workshop Report

CMU/SEI-2001-TR-027
ESC-TR-2001-027

Paul Clements
Patrick Donohoe
Kyo Kang
John McGregor
Linda Northrop

September 2001

Product Line Systems Program

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Norton L. Compton

Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

List of Tables

Abstract

The Fifth Software Engineering Institute Product Line Practice Workshop was a hands-on meeting held in December 2000 to investigate the applicability of product line approaches for families of information systems, to share industry practices in this area, and to evolve the Software Engineering Institute's (SEI's) Framework for Software Product Line Practice. This report synthesizes the workshop presentations and discussions.

1 Introduction

1.1 Why Product Line Practice?

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach for their software systems.

A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

This definition is consistent with the definition traditionally given for any product line—a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission. But it adds more; it puts constraints on the way the systems in a software product line are developed because substantial production economies can be achieved when the systems in a software product line are developed from a common set of assets in a prescribed way. The product line architecture and components are central to the set of core assets used to construct and evolve the products in the product line. This common, product line software architecture[1] capitalizes on commonalities in the implementation of the line of products and provides the structural robustness that makes the derivation of software products from software assets economically viable.

Each product in the product line is formed by taking applicable components from the base of common assets, tailoring them as necessary through preplanned variation mechanisms such

[1] A software architecture of a computing system is the structure or structures of the system that consist of software components, the externally visible properties of those components, and the relationships among them [Bass 98a].

as parameterization or inheritance, adding any new components that may be necessary, and assembling the collection according to the rules of the product line architecture.

By product line practice, we mean the systematic use of software assets to assemble, instantiate, generate, or modify the multiple products that constitute a product line. Building a new product (system) becomes more a matter of assembly or generation than creation. For each software product line there is a predefined guide or plan that specifies the exact product-building approach.

Product line practice involves strategic, large-grained reuse as a business enabler. Some organizations have already experienced considerable savings by using a product line approach for software system production. Other organizations are attracted to the idea but are in varying stages of integrating product line practices.

In January 1997, the Software Engineering Institute (SEI) launched the Product Line Practice Initiative to help facilitate and accelerate the transition to sound software engineering practices using a product line approach. The goal of this initiative is to provide organizations with an integrated business and technical approach to systematic reuse so that they can produce and maintain similar systems of predictable quality more efficiently and at a lower cost.

One of the strategies to reach this goal involves the direct interaction with and nurturing of the community interested in product line practice. This transition strategy has been executed in part by a series of product line workshops organized by the SEI.[1] The workshop described in this report is the fifth such workshop to bring together international groups of leading practitioners from industry to codify industry-wide best practices in product lines. The results of the previous four workshops are documented in SEI reports [Bass 97, Bass 98b, Bass 99, Bass 00]. The SEI has also refined the workshop results through work with collaboration partners, participation in other workshops, and continued research. In addition, the SEI is producing a framework for product line practice. The Framework for Software Product Line Practice written by the SEI describes the foundational product line concepts and identifies the essential activities and practices that an organization must master before it can expect to field a product line of software or software-intensive systems successfully. The framework organizes product line practices into practice areas that are categorized according to software engineering, technical management, and organizational management. These categories do not represent job titles, but rather disciplines. The framework is a living document that is evolving as experience with product line practice grows. Version 3 of the framework was made available on the Web in September 2000 [Clements 00].

---

[1] The SEI also organized the first international Software Product Line Conference (SPLC1) held in Denver, Colorado in August 2000 [Donohoe 00].

Much of the experience that forms the basis for the framework has been derived from software product lines that consist of embedded, hard, real-time systems. Though we had some evidence that showed the success of product line approaches with information systems, we had not focused much attention on this class of systems. This Fifth Product Line Practice Workshop was specifically designed to focus on product line approaches for information systems.

1.2 About the Workshop

The SEI held the fifth in a series of two-day Product Line Practice Workshops in December 2000 to achieve the following goals:

- Investigate the applicability of product line approaches for information systems.
- Share experiences with product line approaches for information systems.
- Stimulate the growth of a network of interest in software product lines.
- Populate the framework with proven product line practices relevant to information systems.
- Identify gaps where experience is not properly reflected in the framework.

The participants in this workshop were invited based upon our knowledge of their company's experience with strategic software reuse through product lines and with information systems. The participants were sent a copy of the SEI's Framework for Software Product Line Practice to provide a common focus to structure the workshop presentations and discussions.

The workshop participants included

- Gary Chastek, SEI
- Sholom Cohen, SEI
- Grant Covell, C-Bridge
- Felix Bachmann, Robert Bosch GmbH
- Len Bass, SEI
- Grady Campbell, SEI
- Paul Clements, SEI
- Patrick Donohoe, SEI
- Robin Getty, BP Oil
- Kyo Kang, SEI
- John McGregor, Korson-McGregor
- Robert Nord, Siemens
- Linda M. Northrop, SEI
- Chris Verhoef, Free University of Amsterdam

Four of our guests had direct experience with software product line approaches for information systems. Each was asked to make a presentation explaining his organization's experience and describing the practices that were used and the issues that were encountered.

On the second day, participant presentations were summarized. Then the participants divided into two working groups to explore the practices and issues surrounding product line approaches for information systems from two angles: software engineering and management. Each working group was asked to begin by characterizing the difference between information systems and other types of systems. The working groups then presented their results to the entire group. The workshop concluded with a discussion of whether the SEI's Framework for Software Product Line Practice was applicable to product lines of information systems.

1.3 About This Report

This report summarizes the presentations and discussions at the workshop. As such, the report is written primarily for product line champions who are already working or initiating product line practices in their own organizations. Those who work in the area of information systems will be especially interested. Technical software managers should also benefit from the information.

The report is organized into four main sections that parallel the workshop format:

1. Introduction
2. Product Line Experiences: Highlights of Participants' Presentations
3. Practices and Issues Related to Product Lines of Information Systems: Working Group Reports
4. Summary

The section following this introduction, "Product Line Experiences: Highlights of Participants' Presentations," synthesizes the product line experience of the workshop participants by identifying themes that emerged from the presentations. Section 3 is composed of the two working group reports. The summary in Section 4 recaps the conclusions of the participants. Additionally, a glossary of terms is provided.

2 Product Line Experiences: Highlights of Participants' Presentations

The workshop began with presentations by four of the invited participants. This section will reflect briefly on the themes that emerged and conclude with valuable insights that were shared.

2.1 Point, Counterpoint Themes

As is the custom with these workshop reports, we do not summarize individual presentations but rather try to draw out common themes articulated by each of the presenters. This product line workshop was a study in fascinating contrasts. Our four outside speakers covered a broad range of situations: one was in the middle of developing a product line, two consult extensively to product line developers, and one has done both. From these wide perspectives, some interesting point/counterpoint themes emerged.

2.1.1 Motivation and Goals

At this workshop, like previous workshops, all of the participants voiced the reasons they have adopted (or counsel their clients to adopt) the product line approach. It is decidedly not because product lines are the trendy thing to do, or even because they will fatten an organization's bottom line. It is because doing business the current way is not an option; disaster is looming. The disaster may be: the inability to turn out planned products; the inability to maintain market share; a mortal threat from a new and very efficient competitor able to offer mass customization that you cannot; or your product suite about to spin out of control because of your inability to handle the complexity of a plethora of almost-alike versions of almost-alike products. "There is no alternative," said one participant. "Running into a wall is a good motivator," agreed another. "We can't continue the way we are," said another. "We won't be in business six months from now [otherwise]," said still another.

While everyone agreed that avoiding disaster was the motivation, they also laid out goals they hoped to reach once they had achieved product line capability. And the goals were all presented as economic. Even if phrased in terms such as "long-term viability" or "market share" or "productivity," economic factors lay at the heart of organizations' product line goals. And yet, the product line champions often do not think in monetary terms at all. Being first to a market was a theme we heard repeated often at this workshop, and while that certainly has an economic incentive associated with it, it's one theme that is very hard to quantify. Similarly, "being agile in the marketplace" and "delivering new solutions quickly" were goals with an economic flavor, but they were hard to measure. This is consistent with other workshop results. One participant at an earlier workshop told us his organization adopted the product line approach for productivity reasons. "We would have tripled our staff overnight if we could have," he said. "Cost was no object. But we couldn't hire that many people; they simply weren't available." The lesson here, echoed at this workshop, is that product lines are about economics, but economics does not usually translate directly to cost.

2.1.2 Hurdles

Product lines are critical to the success of many an organization, and the ones represented at this workshop are no exception. Our speakers made that clear. As noted above, product lines are the escape route from disaster and the key to economic viability. And yet, all of our speakers spoke of daunting and frustrating challenges getting their organizations (or their client organizations) to take the plunge. "Projects have their own agenda," lamented one speaker, recalling the inertia that product-building teams seem to have and the resistance they have to dropping their independence for the greater good. "Politics and indecisiveness get in the way" according to one of the consultants, who viewed management indecisiveness as a major reason why product line efforts fail. Managers, in the experience of many of our guests, rarely put the right incentives in place, an observation that agrees with our own. As long as programmers are rewarded for getting the product out at any cost, even if they shortcut the product line process, that organization has failed to acknowledge with its actions that product lines are critical to its success.

2.1.3 Working in Rarefied Air

All of our participants work or consult to the highest echelons of organizations: as one participant put it, "paying attention to the C*x*Os," where *x* is replaced by your choice of *I* (information), *T* (technical), *O* (operations), or even *E* (executive). Here is where the tough decisions are made, as befits the stratospheric salaries at this level. The buck stops here. And yet, is this really where a product line will succeed or fail? What about the troops? Why should they support a product line approach? What about middle managers? Why should they? How will a product line be launched and institutionalized at the business-unit level, let alone the project-team level? It was not clear how to bridge the gap between motivating the generals and actually putting the troops into action.

2.1.4 Reuse Beyond Code

The concept of software reuse equating to code reuse took a thorough (and richly deserved) beating. All of our participants volunteered that reuse beyond code is where the effort pays off and where they focus their attention. One participant said that his company concentrates on "delivery plus strategy plus education" when servicing its customers. Another participant, trying to bootstrap a product line by rounding up a plethora of already-existing systems across a wide variety of loosely connected business units, said he looks for "any artifact of the design, development, and runtime environments of a system that could reduce the effort or skill required to develop a new system." Code may be in there somewhere, but it certainly is not the primary quarry. For him, the key elements were processes, tools, standards, and frameworks.

And yet, at the end of the day, our participants told us that managers want to see *solutions*—that is, running software. All the standards and processes in the world pale next to a working system. Often, solutions arrive in the form of components, which more often then not consist of code and nothing but code. And so if there is a choice to be made between reusing code and reusing non-code artifacts, it may turn out to be a difficult choice indeed.

2.1.5 Architecture Is Everything—or Not

As at previous workshops, all of our participants presented the underlying architectural concepts (if not actual architectures) behind the work they were doing. The old product line standby, the layered architecture, was still the favorite. One participant said that architecture is the goal and the end state that is sought. Without a common architecture, he said, multi-project initiatives are destined to fail.

And yet, architecture evaluation was not mentioned as a practiced validation step that anyone applied to this most important of all design decisions. More than one speaker mentioned the importance of a community-standard architecture that will drive the financial, banking, insurance, automotive, e-commerce, and a myriad of other industries. But more important than having such an architecture is being the one who gets to define it, thus giving one's own products a mammoth head start in the application field. Who gets to define it? The first organization to arrive with one in hand does. Therefore, the unmistakable yet troubling implication is that it is better to be first with an architecture than to be good with an architecture. Perhaps any architecture will do, as long as it is the first one to be offered up. Architectural issues that arose in this workshop were not of the form "What architecture shall we use?" or "How do architectural styles help us solve our design problems" but rather of the form "Shall we use XML?" or "Shall we choose CORBA or COM?" In other words, technical architectures seemed to enjoy a primacy over what we would call software architecture. And everyone acknowledged a free willingness to "buy" their architecture from their component vendors—that is, adopting whatever architecture is implied by the component technologies they choose.

2.2 Common Themes

Several common themes emerged at this workshop. They include the importance of understanding relevant domains, training and education, making a business case for the chosen strategy, recruiting high-level champions for the product line approach, and banishing indecisiveness.

2.3 Insights

As at previous workshops, the participants left us with a wonderful collection of insights and experiences—knowledge that we can package and share with others. At this workshop, the nuggets included

- from one representative, a clear picture of a large organization migrating to the product line approach, along with its history, motivation, culture, and business climate. To accompany this was a two-pronged strategy to exploit commonality in logic and data, and a layered architecture to take both into account.
- from another representative, a simple but effective mining-to-architecture approach, with metrics to help determine whether an asset belongs in any of three categories: core (used by every product), line of business (used by a subset of the products) or site-specific (used by only a single product). This three-bucket asset repository gives rise to the elegant idea of *release management*, which is releasing new versions of core assets only as necessary so as not to overburden projects with new core assets that they do not need.
- from another representative, the idea of *total delivery* beyond just software, which includes generous quantities of education in the concepts, domain, technologies, and goals
- from another representative, the reminder that what we think of as testing is just a special case of artifact validation, and the reminder that while validation takes time, artifact validation gets you to the end point faster

2.4 Words of Wisdom

A custom has emerged at our product line workshops: we try to capture the best aphorism that our participants bring us. Always, it seems, we learn a new way to express in the most pithy fashion possible a fundamental concept relating to product lines. This year's winner is a four-word injunction about distinguishing between work you are willing to contract to others and work you must do yourself in order to maintain your core competencies: "Don't outsource your brain."

3 Practices and Issues Related to Product Lines of Information Systems: Working Group Reports

Following the presentations, workshop participants divided into two working groups. First each group explored the difference between information systems and other systems. Then one group looked at software engineering practices and the other at management practices, both from the perspective of product lines of information systems. Each group formulated some preliminary conclusions. The following sections contain reports from these two working groups.

3.1 Software Engineering Working Group

The Software Engineering Working Group began with the charge to define what it means to be an information system and, based upon that definition, investigated three questions:

1. What is meant by a product line architecture from an information-systems perspective?
2. What are the component-based trends from an information-systems perspective?
3. What are the product line assets in the information-systems domain?

3.1.1 Definition of Information System

For the purpose of this discussion, an information system was considered to be a software system that gathers, processes, and stores data and transactions related to the purpose of the business. The processing models the workflow of business processes and enforces business rules. The data that is stored is maintained over a longer period of time than for other types of systems. Finally, information systems have fewer physical constraints than do other types of systems, embedded systems in particular.

Examples of information systems include order-taking systems such as those used in e-commerce applications in which a complex workflow is modeled. Even more data-intensive information systems include accounting, banking, brokerage, and insurance claims systems in which large amounts of data must be maintained for a domain-specific life cycle. That life cycle may be: 30 years in the case of a mortgage; in perpetuity in the case of governmental records; or the length of a single transaction.

For purposes of comparison, the software that is an integral part of a cellular telephone or a missile-guidance system is not classified as an information system. Productivity tools such as word processors or spreadsheet applications are not information systems either. These systems process data but in smaller quantities, in smaller chunks, and over shorter time periods. These systems tend to be tools that are used to achieve short-term results. Even other "information systems" such as geographic information systems are outside the context of this discussion.

3.1.2 Product Line Architecture for Information Systems

The issues related to the software architecture for an information system are somewhat different from those for other types of systems. In particular, architects for embedded software systems are more interested in the physical architecture than the logical architecture due to the need to satisfy hardware constraints. Architects of information systems are more concerned with the logical architecture than the physical architecture due to the need to satisfy the logical constraints imposed by workflows and business rules. Lacking hardware constraints, the architects of information systems need other forces to help shape the architecture of the system. Constraints, other than workflows and rules, come in the form of strategic and tactical business goals, the need to interact with other systems, and the physical dispersion of business units. These constraints result in a few standard software architectures that can be followed with some elaboration but little modification.

Information systems are often built on top of standard commercial infrastructures, such as the Common Object Request Broker Architecture (CORBA), Common Object Model (COM) or Java 2 Enterprise Edition (J2EE). The framework may provide: a clear, albeit generic, architectural vision as in CORBA; only a lightweight specification for component interaction as in COM; or a domain-specific solution like J2EE. The selected infrastructure provides a basic framework to which domain-specific structures are added. At the product line level, the domain-specific structures define abstract concepts, such as a claim against an insurance policy. At the product level, these structures are specialized to apply to concrete concepts, such as a disability claim against a health insurance policy.

The framework is populated with components that provide the application-specific behavior. In information systems the components may be created to support specific business goals or processes. By separating these responsibilities, components can be created that are reusable across almost any application, such as application security. Other components can be created that are specific to the product line suite of products, such as a company's basic business rules. Still more specialization results in the components that are used by the individual product teams.

There are a growing number of standard business process definitions for specific industries that result in architectural patterns of interaction between components. For example, the Workflow Management Coalition provides naming conventions, client application programming interfaces, and definitions of XML elements to promote compatibility and data interchange between workflow definitions in different components. These specifications provide sources of structure but also provide a source of abstractions that can lead to the definition of interfaces in the product line architecture. The eXtensible Business Reporting Language (XBRL) is another example of an industry-based effort that defines architectural structures for information systems.

3.1.3 Component-Based Trends in Information Systems

Components that implement stable requirements can quickly reach the maturity needed to become industry standards. Information systems tend to follow similar architectural patterns, and the development of individual systems becomes an exercise in fitting problems to the canonical architecture. This provides an environment in which de facto standards such as Enterprise JavaBeans can gain popularity rapidly.

Components are becoming less dependent on the context in which they are deployed. The goodness of components can be partially determined by how dependent the component is on the services defined by that context. For example, "application servers" are referred to as containers and serve only to forward events to the components that they contain. An entity JavaBean, residing within the application server, usually only needs access to a database to perform its behaviors. The infrastructure chosen as part of the basic architecture establishes one level of context within which components used in the system must fit.

Industry groups are attempting to define standard contexts, such as generic data buses, to support a "Plug 'n Play," component-based construction process. Software product line projects define a standard architecture that constitutes the context within the scope of that product line. This, in turn, promotes the specification of families of components that play specific roles in the standard architecture.

Companies are attempting to influence industry standards by forming "community" efforts to define standards. The Java Community Process is one such effort. These efforts vary in their level of detail and scope. The Java Community Process includes a number of different standards, while the previously mentioned XBRL effort is focused on one standard report language.

3.1.4 Product Line Assets in Information Systems

An asset is anything that you are willing to spend money on to create, make reusable, and maintain. The typical assets of a product line include the requirements, architecture, test cases, and components. For information systems, the specific types of assets include analysis patterns, business rules, data definitions, and workflow definitions.

Analysis patterns that capture the semantic patterns of the business domain are an important asset. They provide a means by which standard problem elements are recognized, captured, and communicated to the individual product teams. These standard patterns of domain information contribute to the identification of standard architectures.

Business rules are first-class entities in information systems. This is due to the fact that the rule sets change often and must be maintained. The rules are an asset because they capture dynamic-control information that is a structure independent of the product line's logical architecture.

Data definitions have long been assets of information systems. The definitions are captured in data dictionaries. In a product line effort, these dictionaries are structured in a hierarchical manner. The most abstract definitions are created and stored at the product line level. The data dictionaries for products contain the more specific definitions and include, but often override, the definitions created at the product line level.

Workflow definitions capture standard business algorithms. The flows for a specific problem domain, such as insurance claim management, are common across applications. These definitions can be captured in interface specifications at the product line level or produced as components and shared among the product teams where specializations of the workflow specifications are applied to tasks.

Information systems have been in existence longer than many other types of systems. As a result, they are likely to have legacy modules that implement specific business functionality that is at the core of an essential process. These assets are mined, packaged, and incorporated in the product line without modification. One would expect that the mining of assets would be critical to the product line effort and provide more of the functionality than in other types of systems.

3.1.5 Conclusions

Product lines of information systems present some unique characteristics, but for the most part follow the same general pattern as other types of product line systems. There are assets specific to information systems, such as business rules and workflow definitions, but they present the same opportunities for incremental specialization between the generic level of a product line and the specialized level of individual products. The architecture of the information system is one of the main assets of a business-information product line effort, but these systems tend to follow more standard patterns and can benefit more from the product line process. These standard architectures lead to a larger component market because there is a larger audience for a given context.

3.2 Management Working Group

The Management Working Group explored management issues that might make adopting a product line approach in the information-systems area different from the embedded-systems area. First the group attempted to establish whether there are any essential distinctions between embedded systems and information systems that would have implications for the adoption or practices of a product line approach. With this discussion as a foundation, the group then considered two issues:

1. whether there are different management issues for a product line of information systems
2. when a product line approach is appropriate for information systems

3.2.1 Distinguishing Information Systems from Embedded Systems

The working group spent significant time exploring system characteristics that might distinguish information systems from embedded systems. As a result, the group identified seven factors, summarized in Table 1, which seem potentially significant. As the group explored example systems, it became clear that the distinction between the two types of systems has blurred in recent years as both categories increase in complexity. While there are factors commonly associated with embedded systems (such as having hard, real-time deadlines or being safety critical) or with information systems (such as an emphasis on a database and extended data persistency), systems of both types are becoming more complex with integrated functionality of both sorts.

For example, an emergency vehicle dispatch system, which uses a global positioning system and a traffic-information system, is an information system that must satisfy real-time deadlines, is safety critical, and has characteristics traditionally associated only with embedded systems. Similarly, modern weapons systems require increasingly complex data management capabilities. While there are remaining distinctions related primarily to hardware control, the group projected that such distinctions will get fuzzier as real-world information systems become larger and more integrated.

Table 1: Factors Distinguishing Embedded Systems from Information Systems

	Embedded System	**Information System**
Data Management	Sometimes	Yes
Hardware Control	Yes	No
Real-Time Deadline	Hard (sometimes)	Soft (sometimes)
Safety Critical	Sometimes	Sometimes
Expertise	Multi-disciplinary	Business functions
Process	Embedded in systems	Top-level business process
Mission Role	Support	Primary

Based on these observations, the group concluded that there are no technical or management issues that are unique to information systems or that would render an "information-systems" product line approach fundamentally different from that for an "embedded-systems" product line. The group also agreed that, irrespective of particular system types, the approach to a product line could vary depending on the product line's size, complexity, and existing legacy software.

3.2.1.1 Product Line Examples that Are Information Systems

As a basis for viewing information systems from a product line perspective, the group identified several instances of systems that seem to have some degree of product line flavor already. For a successful product line, the commonality and variability of products in the product line must be understood, and flexibility must be established in organizational practices to accommodate variabilities. The group suggested several examples of product lines of information systems and identified variabilities that distinguish instance products of each:

- Enterprise Resource Planning (ERP) packages (such as Baan and SAP): business rules, organizational structure, workflow, logistics, laws, and tailoring capabilities
- a package (CARMEN) for complex scheduling and work rosters: laws and schedulable types of resources
- financial information Web sites: information content, analyses, and visual presentation
- Web-hosting provision: load, customer workflow, and business rules
- multi-version, shrink-wrapped software (such as Photoshop): platform, language, feature sets, plug-ins, and allowed number of users

Examining how these products differ, the group observed that product lines of information systems are generally adaptable to factors such as business rules, organizational structures, workflows, workloads, logistics, laws, information content, analysis needs, presentation forms, and computational algorithms used. For example, ERP packages that are deployed internationally must be adapted to local laws and business processes that differ among countries. CARMEN, a resource-scheduling package, must be able to accommodate a wide range of uses and scheduling rules that may cover scheduling airplanes and their pilots as well as scheduling a rail fleet and its crew members.

3.2.2 When to Adopt a Product Line Approach—or Not

The process of deciding whether a product line approach is good for a given organizational enterprise is complex and requires the investigation of many factors. To better understand these factors, the group decided to first explore some of the overriding project conditions that make a product line inappropriate and then to explore some that make one practical.

It is generally accepted that a product line is cost effective when at least three products are to be developed from common assets within some time limit dependent on prevailing product life cycles [Weiss 99]. Therefore, in a situation where an organization plans to develop only one or two systems that are likely to change little over time, a product line may not be an economically viable approach. For example, for systems with a limited use or those that are disposable after a single use (e.g., a data-migration tool for converting a particular company's payroll data from an Informix database to an Oracle database), a product line approach is typically not cost effective.

Also, it takes longer to deploy the first product in a product line because this approach requires the exploration of commonalities and variabilities of products and the development of needed core assets. Therefore, when there is a hard deadline for deploying a mission-critical system, meeting the deadline may be deemed too risky to permit a product line approach for that system.

Conversely, there is a clear benefit to a product line approach when an organization recognizes a future need to build multiple similar products. Additionally, there are project conditions that could render a product line approach economically viable even when there is no expectation of multiple products. For instance, if a product has multiple uses in somewhat different contexts, has multiple installation sites with different operating environments, or is to be released in multiple versions with additional features or enhanced capabilities, a product line approach could improve productivity and be cost effective. In all these cases, expected changes and differences should be identified and properly encapsulated in components.

One of the conditions for a product line is that the organization must be able to characterize the scope of the product line. When its boundaries are unclear, each stakeholder can have different expectations that can lead to failure. Unlike embedded systems with constraints of a physical environment, it is often difficult to establish the boundaries of a product line of information systems. Setting clear boundaries for the product line and then exploring extensions from those boundaries is critical for a successful product line.

3.2.3 Issues in Product Lines of Information Systems

There are several issues particular to product lines of information systems. In the embedded-systems area, a product line approach is usually taken when the producer has a proprietary technology to embed in the product and with a clear goal to either market the product (e.g., cruise-control system for the automobile industry) or deploy the product at multiple sites (e.g., avionics software in every airplane). The group, however, noted that a product line approach is often considered or taken in the information-systems area with only a "perception" of multiple customers or products (i.e., potential as a marketable product). For example, a bank that has developed its own banking system sees other banks doing the same development and then evolves its software into a product. Most banks, however, have their own system already, and the market may not be as real as it was perceived to be.

This example reveals another issue in product lines of information systems. For instance, even when a bank wants to use a banking system developed as a commercial product, it may still want to use its legacy code or other commercial off-the-shelf (COTS) products for the problems unique to its bank. In the information-systems area, a large amount of investment has already been made for both hardware and software by organizations relying on computers for their work, and salvaging this investment complicates their transition to a product line approach.

Another challenge for a product line of information systems is that the context of a product is not tangible, and it is often hard to draw a clear boundary between the product and its contextual entities. In an embedded-systems product line, there is a clearer, tangible boundary between software and the hardware components with which the software interfaces. In a product line of information systems, a product is used within a larger system (e.g., a banking system used in a bank), and the idea of which functions belong where is often different from one customer to another. Also, the product may have to interface with other systems already in place. This intangibility of the product context and the flexibility of the product boundary present a difficult challenge in a product line of information systems.

As more and more information systems are integrated via network (e.g., e-business systems), deciding each user's span of control on common data and functions has become a major problem. In the case of systems for in-house use, the user's span of control can be made clear. In the networked environment, however, many systems with different owners are being integrated, and as these systems interface with each other sharing data and functions, deciding each user's span of control has become a major problem.

3.2.4 Management Differences

The group's discussions suggested two different project contexts in which a product line approach may be taken: proactive (strategic) or reactive. Each of these project contexts presents different management problems.

In a proactive product line, the project has a specific goal to build multiple products for different customers or markets. In this case, the management must have a good marketing strategy with products that meet the needs of different market segments. Also, the product engineers must have an ability to address diverse market needs in a timely and cost-effective way, building products with the right set of core assets. Embedded systems, because of their dependence on long-lead-time hardware engineering and manufacturing investments, often lead to a proactive product line.

In a reactive product line, the project has specific problems to address, and the management has the burden to rationalize that a product line approach is an efficient and effective way to address the problems. For example, if a product line is adopted to address problems maintaining multiple versions of a system, the management must be able to rationalize that a large up-front investment to build high-quality core assets will solve the maintenance problems and eventually yield a long-term cost reduction. Also, when cost reduction is the primary concern, the management must be able to adjust the needs and the cost reduction from the use of COTS products. Information systems most commonly arise in the form of a reactive product line.

3.2.5 Conclusions

As systems of all types become more complex and integrated, there will no longer be system characteristics that are unique only to embedded systems or to information systems. Some embedded systems (e.g., missile-guidance systems with terrain-map information) already use databases while some information systems (e.g., emergency vehicle dispatch systems) have real-time and safety requirements. As such systems are integrated to exchange and share information, it will become even more difficult to draw a line between the two types. The group concluded that a sound product line approach must be able to accommodate both types of system characteristics and variabilities.

4 Summary

The SEI's Fifth Product Line Practice Workshop explored the applicability of product line approaches for information systems. Though the group attempted to distinguish information systems from other systems, we found more similarities than differences. The boundary between the two is so hazy we were tempted to conclude that the term "information system" (along with whatever other terms it is meant to be in contrast with, most notably "embedded system") is a holdover from earlier days in computing.

The lesson for software product lines is that there does not seem to be a large chasm between product line methods for information systems and product line methods for other kinds of systems. Indeed the processes, techniques, and topics covered by the SEI's Software Product Line Framework are applicable to the construction of information systems. The topics covered in the framework stress the ideas of commonality, architecture, and multilevel planning. Information systems have historically followed a few very standard approaches so that systems exhibit a high degree of commonality. Each of these standard approaches includes a well-defined reference architecture from which the application architecture is derived. The hierarchical nature of many business organizations matches the organization of a product line organization. The definitions of multiple levels of responsibilities in a product line's product plan are compatible with the typical business organization.

The results of this workshop will be incorporated into the framework,[1] which will continue to be refined and revised as the technology matures and as we continue to receive feedback and to work with the growing community of software engineers championing a product line approach. If you have any comments on this report, if you have experience with product lines of information systems that you would like to share, or if you are using a product line approach in the development of software-intensive systems and would like to participate in a future workshop, please send electronic mail to Linda Northrop at lmn@sei.cmu.edu.

[1] Version 3 of the SEI's Product Line Practice Framework is available on the Web at: http://www.sei.cmu.edu/plp/framework.html. The contents of Version 4 of the framework form Part II of *Software Product Lines: Practices and Patterns* [Clements 01].

References

[Bass 97] Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; & Withey, James. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003, ADA327610). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. Available WWW: <http://www.sei.cmu.edu/publications/documents/97.reports/97tr003/97tr003abstract.html>.

[Bass 98a] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice*. Reading, MA: Addison-Wesley Longman, Inc., 1998.

[Bass 98b] Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; Smith, Dennis; & Withey, James. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015, ADA343688). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available WWW: <http://www.sei.cmu.edu/publications/documents/98.reports/98tr015/98tr015abstract.html>.

[Bass 99] Bass, Len; Campbell, Grady; Clements, Paul; Northrop, Linda; & Smith, Dennis. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003, ADA361391). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW: <http://www.sei.cmu.edu/publications/documents/99.reports/99tr003/99tr003abstract.html>.

[Bass 00] Bass, L.; Clements, P.; Donohoe, P.; McGregor, J.; & Northrop, L. *Fourth Product Line Practice Workshop Report* (CMU/SEI-2000-TR-002, ADA375843). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. Available WWW: <http://www.sei.cmu.edu/publications/documents/00.reports/00tr002.html>.

[Clements 00]	Clements, Paul; Northrop, Linda; et al. *A Framework for Software Product Line Practice, Version 3* [online]. Available WWW: <http://www.sei.cmu.edu/plp/framework.html> (August 2000).
[Clements 01]	Clements, Paul & Northrop, Linda. *Software Product Lines: Practices and Patterns.* Reading, MA: Addison-Wesley, 2001.
[Donohoe 00]	Donohoe, P., ed. *Software Product Lines: Experience and Research Directions.* Boston, MA: Kluwer Academic Publishers, 2000.
[Weiss 99]	Weiss, David M. & Lai, Chi Tau Robert. *Software Product Line Engineering*. Reading, MA: Addison-Wesley, 1999.

Glossary

Application engineering	An engineering process that develops software products from partial solutions or knowledge embodied in software assets
Core asset	A software artifact that is used in the production of more than one product in a software product line. A core asset may be an architecture, a software component, a process model, a plan, a document, or any other useful result of building a system.
Domain	An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area
Domain analysis	Process for capturing and representing information about applications in a domain, specifically common characteristics and reasons for variability
Domain engineering	An engineering process that develops software assets for one or more domains
Economies of scale	The condition where fewer inputs such as effort and time are needed to produce greater quantities of a single output
Economies of scope	The condition where fewer inputs such as effort and time are needed to produce a greater variety of outputs. Greater business value is achieved by jointly producing different outputs. Producing each output independently fails to leverage commonalities that affect costs. Economies of scope occur when it is less costly to combine two or more products in one production system than to produce them separately.
Platform	A core software asset base that is reused across systems in the product line

Practice area	A body of work or a collection of activities that an organization must master to successfully carry out the essential work of a software product line
Product line	A group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission area
Product line approach	A system of software production that uses software assets to modify, assemble, instantiate, or generate a line of software products
Product line architecture	Description of the structural properties for building a group of related systems (i.e., product line), typically the components and their interrelationships. The inherent guidelines about the use of components must capture the means for handling required variability among the systems. (Sometimes called a reference architecture)
Product line scope	Description of the products that will constitute the product line
Product line system	A member of a software product line
Production plan	The guide to how products in the software product line will be constructed from the product line's core assets
Software architecture	The structure or structures of the system, which consists of software components, the externally visible properties of those components, and the relationships among them
Software product line	A set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE September 2001	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Fifth Product Line Practice Workshop Report		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Paul Clements, Patrick Donohoe, Kyo Kang, John McGregor, Linda Northrop		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2001-TR-027
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2001-027
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE
13. ABSTRACT (MAXIMUM 200 WORDS) The Fifth Software Engineering Institute Product Line Practice Workshop was a hands-on meeting held in December 2000 to investigate the applicability of product line approaches for families of information systems, to share industry practices in this area, and to evolve the Software Engineering Institute's (SEI's) Framework for Software Product Line Practice. This report synthesizes the workshop presentations and discussions.		
14. SUBJECT TERMS commercial product line practice, DoD product line practice, Software Product Line Practice Framework, product line workshop, software asset, software architecture, software product lines		15. NUMBER OF PAGES 34
16. PRICE CODE		

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL